An Acceleration Scheme for Mini-batch, Streaming PCA

Salaheddin Alakkari
AlakkarS@tcd.ie
John Dingliana
John.Dingliana@scss.tcd.ie
School of Computer Science and Statistics,
Trinity College Dublin,
Dublin, Ireland

Abstract

In this paper, we propose an acceleration scheme for mini-batch streaming PCA methods that are based on the Stochastic Gradient Approximation. Our scheme converges to the first $k > 1$ eigenvectors in a single data pass even when using a very small batch size. We provide empirical convergence results of our scheme based on the spiked covariance model. Our scheme does not require any prior knowledge of the data distribution and hence is well suited for streaming data scenarios. Furthermore, based on empirical evaluations using the spiked covariance model and large-scale benchmark datasets, we find that our acceleration scheme outperforms related state-of-the-art online PCA approaches including SGA, Incremental PCA and Candid Covariance-free Incremental PCA.

1 Introduction

Principal Component Analysis (PCA) is one of the most important machine learning and dimensionality reduction techniques. It is an unsupervised learning model that captures the maximal variability of an input data using a lower dimensional space. PCA has attracted research in many different scientific fields for the last three decades. It also forms the cornerstone for developing and understanding many AI techniques, specifically in the area of Neural networks.

The main task of PCA is as follows: Given n data samples $\{x_i\}_{i=1}^n$ where each sample is centered and lies in R^d, PCA finds an orthogonal low dimensional basis where each sample can be expressed as weighted sum of these basis vectors with minimal squared error [7]. These basis vectors correspond to the eigenvectors of the covariance matrix $C = \frac{1}{n-1}\sum_{i=1}^n x_i x_i^T$ which can be obtained by solving the secular equation

$$(C - \lambda I)\, v = 0; v^T v = 1, \tag{1}$$

where $v \in R^d$ is the eigenvector and $\lambda \in R$ is its corresponding eigenvalue. The resulting eigenvectors are sorted in descending order based on their eigenvalues. In practice, only a small number of eigenvectors, $k \ll n$, are chosen to form the eigenspace, namely those associated with the higher eigenvalues. Equation 1 can be solved by applying Eigenvalue Decomposition (EVD) on the corresponding covariance matrix C (or dual covariance matrix $C_{dual} = \frac{1}{d-1}X^T X$). EVD requires $\mathcal{O}(nd\min(n,d))$ FLOPs of computation and

$\mathcal{O}(nd + \min(n,d)^2)$ memory space for storing the covariance matrix. When n and d are large, one can get rid of computing and storing the covariance matrix by applying Singular Value Decomposition (SVD) directly to the input samples. However, the computational cost remains $\mathcal{O}(nd\min(n,d))$ FLOPs for SVD. Although such standard approaches give the best quality performance in terms of minimizing the loss function, handling datasets of very large number of samples, n, or dimensionality, d, becomes infeasible due to the quadratic complexity dependence on the data size.

Many algorithms have been developed to find the optimal eigenvectors with lower space and time complexity. We can classify each of these algorithms into two main categories: offline and online algorithms. Offline techniques compute the optimal eigenvector using a number of iterations where at each iteration a single pass over the entire dataset is performed. While using these algorithms requires the presence of all samples, they usually provide excellent convergence after very few iterations. Online approaches (also referred to as memory-limited or streaming approaches), on the other hand, aim to provide an acceptable convergence after only a single pass over the entire dataset. Thus, they are more appropriate when dealing with streaming data or when data samples are too large to fit into the memory space. While an online approach provides potential solutions to a vast number practical use cases, existing online algorithms cannot be applied efficiently in many of these scenarios for two main reasons. Firstly, memory limitations may significantly affect their convergence. Secondly, many of these algorithms are parametric and hence lead to an ensuing parameter-tuning problem. In many cases, the optimal settings require finding other statistical properties using additional pre-processing passes over the data which means violating the online condition.

Considering a dataset of stationary distribution $X = [x_{t_1}, x_{t_2}, \ldots, x_{t_n}] \in R^{d\times n}$ where d is the total number of dimensions (attributes), our goal in this paper is to find the top k eigenvectors from a single pass of the data without any prior knowledge about the distribution of input samples. We propose an acceleration scheme for streaming PCA algorithms that are based on Stochastic Gradient Approximation (SGA). Such methods enjoy solid theoretical guarantees but may be very slow in convergence due to unsuitable choices of learning rate or initialization. Our approach aims to accelerate such algorithms in the mini-batch setting, where each update step processes multiple samples of the data that are visited only once. Such a mode of learning has many advantages over traditional online techniques, which process only one sample per update. Mainly, processing samples in a mini-batch manner increases speed because of the reduction in number of expensive orthogonalization steps in addition to reducing overhead and increasing parallelism. Our work is an extension to the mini-batch power method proposed by Mitliagkas et al. [12]. We conducted empirical evaluations using the spiked covariance model and other large-scale benchmark datasets. Results show that our approach outperforms state-of-the-art online PCA algorithms including Stochastic Gradient Approximation (SGA), Incremental PCA (IPCA), Candid Covariance-free Incremental PCA (CCIPCA) and Similarity Matching (SM).

Main Contributions

- We extend the definition of state-of-the art streaming PCA methods by studying their performance in the mini-batch setting which is advantageous over the traditional one-sample-per-update mode in terms of run time.

- We propose an acceleration scheme for SGA-based methods in the mini-batch setting

and show that such acceleration outperforms state-of-the-art techniques.

2 Review of State-of-the-art Streaming PCA Methods

Among all streaming PCA approaches, the Stochastic Gradient Approximation (SGA) is one of the oldest attempts to address the online learning behaviour of PCA. Such approaches are based on the assumption that the covariance matrix can be approximated using an arbitrary sample x from the input dataset $E\left(xx^T\right) = C$. The most well-known SGA techniques are the ones proposed by Krasulina [8] and Oja [13, 14]. The update rule according to Oja's method is as follows $W_{t+1} = \frac{W_t + \eta_t x_t x_t^T W_t}{\left\|W_t + \eta_t x_t x_t^T W_t\right\|}$ where the learning rate should obeys Robbins-Monro conditions ($\sum_t^\infty \eta_t = \infty$ and $\sum_t^\infty \eta_t^2 < \infty$). A typical form of the learning rate is $\eta_t = c/t$ for some constant c. One main limitation of these techniques is their performance sensitivity to different choices of learning rate, which may significantly affect the convergence rate. According to Balsubramani et al. [2] the optimal performance of SGA is achieved when $c = \frac{a}{\Delta}$ where $a > 1$ and $\Delta = \lambda_1 - \lambda_2$ is the eigengap. Hence finding the optimal learning rate becomes infeasible in case of streaming scenarios as computing the eigengap requires a pre-processing data pass. Furthermore, this suggests that each individual eigenvector (when computing $k > 1$ eigenvectors) should have its own learning rate. The computational cost of these algorithms is $\mathcal{O}(kd)$ FLOPs where k is the number of eigenvectors to extract and d is the dimensionality. These techniques are considered the cheapest in terms of computational cost and space complexity. Generalization of these techniques for extracting multiple eigenvectors ($k > 1 case$) can be achieved as $W_{t+1} = \text{Orthogonalize}\left(W_t + \eta_t x_t x_t^T W_t\right) \in R^{d\times k}$ where orthogonalization is done either using Gram-Schmidt orthogonalization which requires $\mathcal{O}(k^2 d)$ extra FLOPs or using deflation which costs $\mathcal{O}(kd)$ FLOPs but may fall in round-off errors producing vectors that are not fully orthogonal. It can be proven that the orthogonalization step is independent of convergence performance. Hence, one does not need to apply the Gram-Schmidt process at each update but rather we can perform this after every B number of updates. Other approaches that are based on SGA include the mini-batch (or block) power method (proposed by Mitliagkas et al. [12]) which applies the power iteration on a mini-batch of the data that is passed only once and will never be revisited again. The main drawback of this method is that it may not converge in case of very small mini-batch sizes B. They proved the convergence of their method when $B \geq n/\log(d)$. Candid Covariance-free Incremental PCA (CCIPCA) [17] is another streaming PCA method that has a major advantage over Oja's technique in the fact that it is parameter-free. Other parameter-free streaming PCA methods include Incremental PCA (IPCA), which is based on diagonalizing a reduced order matrix of size k instead of dealing with the massive $d \times d$ covariance matrix [1]. The main downside of IPCA is that it cannot be applied in mini-batch mode. According to Cardot et al. [3] when comparing between, SGA, IPCA and CCIPCA, it was found that IPCA and CCIPA produced the best convergence results. Recently, Pehlevan et al. proposed a method for finding top k eigenvectors and their corresponding scores that works in both online and offline modes [15] which is based on the similarity matching objective function. Excluding the mini-batch power method, the performance of Oja, CCIPCA and SM in the mini-batch mode is a matter for further research.

3 An Acceleration Scheme for Mini-batch SGA

In this section, we describe our acceleration scheme. The main idea of our acceleration is to catalyze the convergence of streaming PCA by seeking a steady state in which successive updates become identical. Such a steady state corresponds to the optimal solution. More technically, assuming that the acceleration function $g(W_{t+1}, W_t, t)$ takes as an input the updated eigenvectors at times t and $t+1$ and the number of current update, the main objective is to maximize the following function

$$G(W_{t+1}, W_t) = W_{t+1}^T W_t W_t^T W_{t+1}$$

where W_t and W_{t+1} are normalized and lie in R^d space. Hence, the objective function targets a maximum value of one. This maximization satisfies an important condition of convergence for all online PCA algorithms. By directly applying the gradient ascent rule on the objective function, one gets

$$\begin{aligned} g(W_{t+1}, W_t, t) &= W_{t+1} + \frac{\alpha_t}{2} \frac{\partial}{\partial W_{t+1}} G(W_{t+1}, W_t) \\ &= W_{t+1} + \alpha_t W_t W_t^T W_{t+1} \\ &= \left(I + \alpha_t W_t W_t^T\right) W_{t+1} \end{aligned}$$

where α_t is the learning rate, the values for which we will justify later in this study. This leads to the general update rule

$$W_{t+1} = \frac{\tilde{W}_{t+1} + \alpha_t W_t W_t^T \tilde{W}_{t+1}}{\left\| \tilde{W}_{t+1} + \alpha_t W_t W_t^T \tilde{W}_{t+1} \right\|}, \tag{2}$$

where $\tilde{W}_{t+1} = f\left(W_t, \tilde{X}_t\right)$ is the eigenvector estimation based on the update method f that we wish to accelerate. Here, $\tilde{X}_t = \{x_i\}_{i=Bt+1}^{B(t+1)}$ is a subset of the input dataset revealed at time t with cardinality B and $W_t \in R^d$ is the recent updated eigenvector. We call this subset a block of samples of size B. For instance, in case of Oja's rule $f\left(W_t, \tilde{X}_t\right) = W_t + \eta(t) \tilde{X}_t \tilde{X}_t^T W_t / B$ and in case of block power $f\left(W_t, \tilde{X}_t\right) = \tilde{X}_t \tilde{X}_t^T W_t / B$. Note that here we consider a block variant of Oja's method, where at each iteration we update the eigenvector based on a block of recent time-steps instead of using only the most recent one. Our scheme can be easily generalized to extract multiple eigenvectors ($k > 1$ case) by defining $W_t \in R^{d \times k}$ and replacing the normalization process by the Gram-Schmidt orthogonalization or deflation process. It is also worth mentioning that there are two main situations in which the objective function can be maximized. The first situation is when subsequent updates are converging towards the optimal solution which is the desired behaviour that we wish to get. The second case is when subsequent eigenvectors are not significantly changing due to the very small step-size when accelerating Oja's method. Therefore when accelerating Oja's rule, it would be better to set the learning rate with larger values.

Choices for Learning Rate α_t

We now investigate the appropriate learning rate α_t for our acceleration scheme. In general, we assume that such a learning rate increases with time and has the form $\alpha_t = \mathcal{O}(t)$. This is a somewhat trivial choice since one would expect that as t increases W_t should give a better

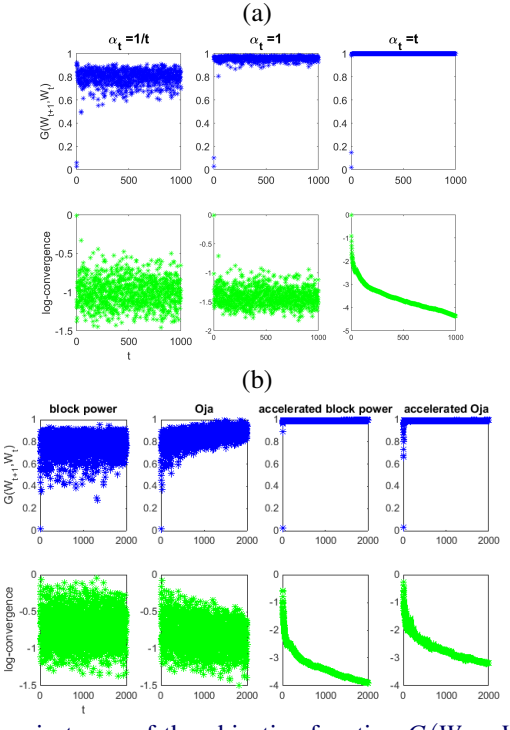

Figure 1: Performance in terms of the objective function $G(W_{t+1}, W_t)$ (Top) and the log-convergence defined as $\log_{10}\left(1 - \frac{\|X^T W_t\|_F^2}{\|X^T V^*\|_F^2}\right)$ (Bottom) when computing the first eigenvector of the spiked covariance model according to [12]. (a) Accelerated block power using decreasing, constant and increasing learning rates. (b) Perfromance of Oja and block power before and after applying our acceleration using increasing learning rate.

estimation of the optimal solution. In other words, we can think of the learning rate in this case as a certainty factor of the accuracy of W_t. Figure 1 (a) shows the performance when accelerating the block power using decreasing $\alpha_t = 1/t$, constant $\alpha_t = 1$, and increasing $\alpha_t = t$ learning rates where it is obvious that increasing the learning rate provides the best performance. The use of increasing learning rates for training, while not usual, has recently been found to provide much better convergence rates for Neural Networks [16]. Comparisons of convergence rates for the block power and Oja's methods before and after applying our acceleration scheme are shown in Figure 1 (b). It is evident that using our scheme significantly enhanced the convergence rates. In addition, one can also note that the convergence rates are achieved when the objective function reaches its optimal value of one. We will consider two strategies for setting the learning rate α_t. The first strategy that we propose is setting $\alpha_t = \frac{t}{1+cz_t}$ where c is some small constant (say 1) and $z_t \in [0,1]$ is random variable of uniform distribution. The second strategy is based on setting $\alpha_t = \frac{t}{1+cz_t/t}$ where c in this case takes larger values depending on the sample size (say 1,000 in case the number of samples is of the order of thousands). The main motivation for employing such a stochastic paradigm is to allow for some self-adaptive behaviour in finding the optimal learning path. In the next section we will test the technique using both strategies.

4 Performance Evaluation

In this section, we will evaluate the aforementioned algorithms from different perspectives. In general, evaluating online PCA approaches is challenging especially when computing $k > 1$ eigenvectors. On one hand, we need to measure convergence to a global solution which is found by computing the traditional SVD in the batch mode. Hence, such measurement would not be possible if the dataset is too large to perform the SVD. Also in many cases, the top eigenvectors are very similar in terms of their eigenvalues leading to a phenomenon where the online algorithm produces a rotated eigenspace of eigenvectors that differ from the original ones. Furthermore, the performance of specific online PCA approaches might vary significantly for different trials of exactly the same parameter settings due to random initialization. In our study, we use the following function to evaluate convergence $\log_{10}\left(1-\frac{\|X^T W_t\|_F^2}{\|X^T V^*\|_F^2}\right)$ where $\|.\|_F^2$ is the squared Frobenius norm and $V^* \in R^{d\times k}$ are the optimal k eigenvectors computed using batch PCA. In order to limit variation in performance, we run 10 trials and report the average performance of each method.

4.1 Complexity and Parameter Dependence

The first important quality factor to compare is the space and time complexities and the parameter dependence of each method. Table 1 summarizes the space and time complexity per sample per update of each method with their parameter dependence. We did not consider the orthogonalization process in the complexity analysis of SGA methods (Oja's and mini-batch power) since one does not need to perform it for each update. The fully online column indicates the methods that process one sample per update. One can note that CCIPCA enjoys the cheapest space and computational cost in addition to being parameter-free and generalizable to work in mini-batch (block) manner. IPCA is rather more expensive in terms of computational cost by a factor of k due to the SVD operation involved on the reduced order matrix Q_n. As we discussed earlier, Oja's method, despite its very well established theoretical analysis and low computational cost, is not practical because of its performance sensitivity to parameter change. While our algorithm is still parametric, it is capable of finding the optimal solution when using the strategies that we proposed and hence is less sensitive to parameter change.

Table 1: Summary of online PCA methods in terms of complexity and parameter dependence.

Method	time complexity	space complexity	mini-batch	fully online	parameter-free	parameter-sensitivity
Oja	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✓	✗	high
mini-batch power	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✗	✓	-
CCIPCA	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✓	✓	-
IPCA	$\mathcal{O}(k^2 d)$	$\mathcal{O}(kd)$	✗	✓	✓	-
SM	$\mathcal{O}(k^3+kd)$	$\mathcal{O}(kd)$	✓	✓	✗	low
accelerated SGA	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✗	✗	low

4.2 Convergence Analysis on Spiked Covariance Model

We test each method on synthetic datasets generated using the spiked covariance model based on [12] where each time-step is drawn from the following generative model

$$x_i = Az_i + \sigma N_i,$$

where $A \in [-1,1]^{d\times k}$ is a fixed matrix, $z_i \in R^k$ is a random weight vector based on standard normal distribution and $\sigma N_i \in R^d$ is a Gaussian noise vector of standard deviation σ. The task is to restore the component matrix A from the noisy samples x_i. We test the online techniques for the following settings $k = 10$, $d = 1000$, $n = 10,000$ and $\sigma = 1$. In such settings, the first k eigenvectors will be very close to each other in terms of their eigenvalues resulting in very small eigengaps. We would like to test the online methods for recovering the first $q = 5$ eigenvectors instead of the the full 10 vectors. In practice, one does not always know the exact number of eigenvectors embedded in the model. Rather, such a choice is highly motivated by computational capabilities. For mini-batch methods, we set the block size to $B = 10$ and $B = 100$ in order to study convergence when using a very small block (mini-batch) size. We ran our acceleration using the two strategies proposed. This allows for better performance analysis by considering our acceleration scheme as a stochastic process. All methods were initialized using the same random vectors from a unit sphere.

Table 2 shows the convergence results of each method. One can note that our acceleration outperforms other competitors and significantly enhanced the performance of block power and Oja's method. Despite setting different learning rates for Oja's method, it results in relatively poor quality performance. In general, increasing the mini-batch size for SGA methods results in better performance unlike similarity matching and CCIPCA where there seems no clear relation between the block size and convergence. In terms of the fully offline methods, IPCA converges better than CCIPCA, SM and Oja's method. Table 3 reports the run time of Similarity Matching, Incremental PCA and our acceleration technique to get a better idea of the speedup gained when using the mini-batch mode. The algorithms were run on a machine with 3.7 GHz Intel Xeon CPU. Due to reduced order SVD involved, IPCA takes the longest time compared to our acceleration technique and SM which is consistent with the computational complexity analysis of these approaches. The accelerated mini-batch SGA when using batch size $B = 10$ has similar run time to SM which is because of the Gram-Schmidt orthogonalization involved. However, when using a larger batch size of $B = 100$ the run time was significantly reduced to 0.19 seconds in the $k = 5$ case and 0.4 in the rank $k = 10$ case. Figure 2 shows the convergence rates based on one trial where it can be noted that all mini-batch methods process data in much lower run-time compared to online methods.

Table 2: Convergence of each method to the first five eigenvectors of the spiked covariance model generated with $k = 10$.

Method	**fully online**	**B=10**	**B=100**
Oja $\eta_t = \frac{1}{t}$	-1	-1.03	-1.46
Oja $\eta_t = \frac{10}{t}$	-0.77	-0.8	-1.44
Oja $\eta_t = \frac{100}{t}$	-0.7	-0.75	-1.43
mini-batch (block) power	-	-0.75	-1.43
CCIPCA	-1.59	-1.77	-1.54
IPCA	-1.7	-	-
SM	-1.04	-0.68	-0.65
accelerated Oja (1st strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	-	-1.77	-1.56
accelerated Oja (2nd strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000\times z_t/t}$	-	$\mathbf{-2.27}$	$\mathbf{-2.72}$
accelerated block power (1st strategy) $\alpha_t = \frac{t}{1+z_t}$	-	-1.86	-1.56
accelerated block power (2nd strategy) $\alpha_t = \frac{t}{1+1000\times z_t/t}$	-	-1.87	-2.64

Table 3: Run time (in seconds) of SM, IPCA and our acceleration after a single pass on the spiked covariance model with $d = 1,000$ and $n = 10,000$.

	SM	**IPCA**	**accelerated SGA** $B = 10$	**accelerated SGA** $B = 100$
$k = 5$	0.93	1.33	0.75	**0.19**
$k = 10$	1.65	2.46	1.69	**0.4**

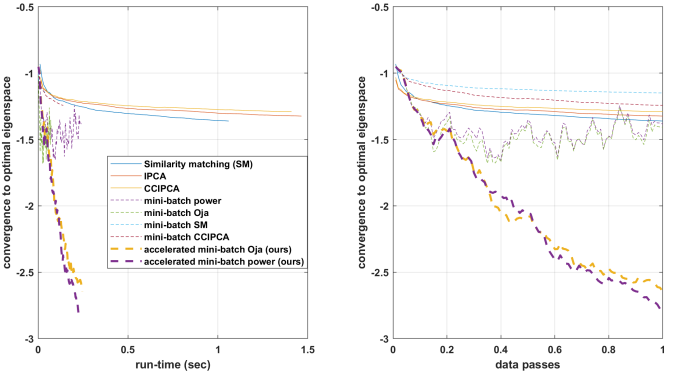

Figure 2: Convergence vs. run-time and convergence vs. data pass when computing the first five eigenvectors of the spiked covariance model.

4.3 Convergence Analysis on MNIST Dataset

We also analyze convergence of each method on the MNIST dataset which consists of $70,000$ grayscale images of hand-written digits each of resolution 28×28 [10]. The MNIST dataset is widely used as a benchmark dataset in machine learning applications. To get a better idea of the performance of the online methods in noisy conditions, we further test the online methods on a noisy version of the dataset. Figure 3 (a) and (b) show several samples before and after adding Gaussian noise of standard deviation $\sigma = 1$. Interestingly, we noticed that in such noisy settings, the batch PCA perfectly recovers the optimal $k = 5$ eigenvectors. Table 4 shows the convergence of each method to the first $k = 1$ eigenvector and $k = 5$ eigenvectors after a single data pass. For the mini-batch (block) modes, we set the block-size to $B = 100$. As we mentioned earlier, we run 10 trials for each method and then report the mean and standard deviation. In all cases, our acceleration provides the best results especially when applying our second strategy for setting the learning rate ($\alpha_t = \frac{t}{1+1,000\times z_t/t}$). The mini-batch power method without acceleration, on the other hand provides the worst results especially in the noisy conditions where convergence does not go below -0.24.. Figure 3 (c) shows the the recovery of the first three eigenvectors of the noisy MNIST dataset using each method. One can note that our acceleration and IPCA perform well at recovering the first three eigenvectors. On the other hand, similarity matching and CCIPCA, despite their very good convergence results, produce a rotated eigenspace with eigenvectors differing from the optimal solution. The code files for running MNIST and Spiked Covariance Model experiments are provided as supplementary material.

4.4 Larger-scale Benchmark Datasets

Up to this stage, all the experiments have been conducted on datasets that are small enough to acquire the optimal solutions using batch PCA. In this section, we will study the performance of streaming PCA methods on larger-scale datasets where applying the batch PCA is not feasible. Moreover, many of the datasets that we study cannot fit into the RAM space of an average machine. Table 5 includes a summary of the datasets investigated in this section. The Labeled Faces in the Wild (LFW) dataset is one of the most well-known face

Table 4: Convergence of each method to the first eigenvectors of the MNIST dataset.

	MNIST		noisy MNIST	
Method	k=1	k=5	k=1	k=5
mini-batch (block) power	-1.19 ± 0.0	-1.086 ± 0.0	-0.24 ± 0.2	-0.22 ± 0.011
CCIPCA	-2.3 ± 0.0	-2.22 ± 0.49	-1.9 ± 0.46	-2.04 ± 0.3
mini-batch CCIPCA	-2.44 ± 0.0	-2.45 ± 0.58	-2.2 ± 0.41	-1.89 ± 0.3
IPCA	-2.47 ± 0.0	-2.64 ± 0.003	-2 ± 0.5	-2.02 ± 0.25
SM	-2.55 ± 0.09	-3.15 ± 0.35	-2.35 ± 0.75	-2.16 ± 0.33
mini-batch SM	-2.08 ± 0.65	-2.16 ± 0.6	-1.96 ± 0.82	-1.83 ± 0.29
accelerated Oja (1^{st} strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	-2.45 ± 0.02	-2.32 ± 0.48	-2.1 ± 0.49	-1.85 ± 0.26
accelerated Oja (2^{nd} strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000 \times z_t/t}$	**-3.4 ± 0.14**	**-3.7 ± 0.16**	**-2.87 ± 0.07**	**-2.6 ± 0.1**
accelerated block power (1^{st} strategy) $\alpha_t = \frac{t}{1+z_t}$	-2.7 ± 0.04	-2.6 ± 0.6	-2.42 ± 0.5	-2.04 ± 0.3
accelerated block power (2^{nd} strategy) $\alpha_t = \frac{t}{1+1000 \times z_t/t}$	**-3.6 ± 0.12**	**-3.88 ± 0.07**	**-2.86 ± 0.04**	**-2.66 ± 0.17**

datasets consisting of over 13,000 face images of 1,680 subjects (most subjects are famous politicians). The CelebA face dataset is similar to LFW but much larger-scale with over 200,000 images of 10,177 celebrities. Computing colored eigenfaces for such large-scale face datasets is not widely discussed in the literature due to the significant extra dimensionality (and hence extra computational cost) caused when adding the color channels. The significant eigenfaces produced using each method are provided in the appendix. The LSUN bedroom dataset (part of the Large-scale Scene Understanding challenge) is the newest, but largest in scale. All studied datasets contain images of highly uncontrolled environments which presents an added challenge to the task of pattern extraction. Since we cannot know the optimal solution using batch PCA, the convergence will be measured in terms of the average Mean-Squared-Error (MSE) between original samples and reconstructed ones after computing the first significant eigenvectors using each technique. We compute the first five eigenvectors using each technique and compare the results. Table 6 shows the MSE achieved by each method after a single pass over each dataset. One can note that in all experiments, our second acceleration strategy produces the lowest errors. In addition, it can be noted that extending SM and CCIPCA to work in mini-batch mode does not always result in better accuracy. Amongst all techniques, the mini-batch power results in the highest errors due to the relatively small size of mini-batches.

Table 5: Summary of the large-scale datasets used in the experiments.

	LFW faces	CelebA faces	LSUN bedroom
no. of samples	13,233	202,599	3,033,042
sample type	250×250 RGB images	178×218 RGB images	256×256 RGB images
Reference	[9, 8, 6, 9]	[10]	[33]

Table 6: MSE achieved by each method when computing the first five eigenvectors after a single pass on the corresponding dataset.

Method	**LFW faces**	**CelebA faces**	**LSUN bedroom**
mini-batch (block) power	0.042395	0.038	0.040677
CCIPCA	0.040468	0.0363	0.038781
mini-batch CCIPCA	0.040478	0.0366	0.038768
IPCA	0.040462	0.0368	0.038796
SM	0.041548	0.0368	0.038777
mini-batch SM	0.040488	0.0366	0.038771
accelerated Oja (1^{st} strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	0.040488	0.0363	0.038766
accelerated Oja (2^{nd} strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000 \times z_t/t}$	**0.040412**	**0.0362**	**0.038758**
accelerated block power (1^{st} strategy) $\alpha_t = \frac{t}{1+z_t}$	0.040487	0.0363	0.038768
accelerated block power (2^{nd} strategy) $\alpha_t = \frac{t}{1+1000 \times z_t/t}$	**0.040408**	**0.0362**	**0.038758**

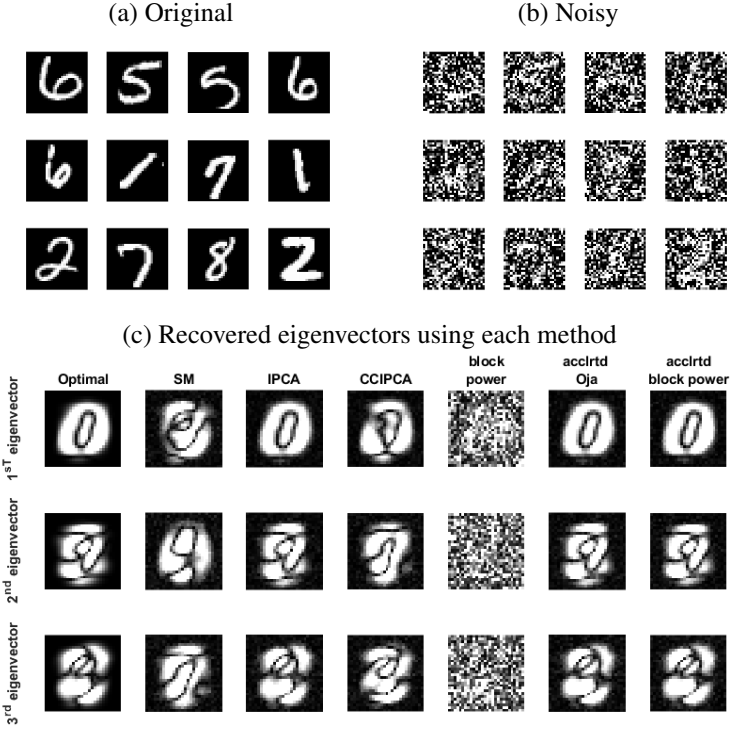

Figure 3: Recovery of the first three eigenvectors of the noisy MNIST using each method.

5 Discussion

In this paper, we proposed an acceleration scheme for SGA-based streaming PCA methods. Particularly, our approach aims to accelerate such algorithms in the mini-batch mode, where each update step processes multiple samples of the data. Experiments on the spiked covariance model, MNIST dataset and large-scale benchmark datasets showed that applying our scheme to accelerate Oja's method and mini-batch power using the second learning strategy outperformed state-of-the-art streaming PCA algorithms. While our scheme was shown to converge very well using stochastic strategies for setting learning-rates, investigating other strategies to further improve the performance is an interesting topic for further research.

6 Acknowledgements

This research has been conducted with the financial support of Science Foundation Ireland (SFI) under Grant Numbers 13/IA/1895 and 13/RC/2106.

References

[1] Raman Arora, Andrew Cotter, Karen Livescu, and Nathan Srebro. Stochastic optimization for PCA and PLS. In *Communication, Control, and Computing (Allerton), 2012*

50th Annual Allerton Conference on, pages 861–868. IEEE, 2012.

[2] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The fast convergence of incremental PCA. In *Advances in Neural Information Processing Systems*, pages 3174–3182, 2013.

[3] Hervé Cardot and David Degras. Online principal component analysis in high dimension: Which algorithm to choose? *International Statistical Review*, 86(1):29–50, 2018.

[4] Gary B. Huang, Vidit Jain, and Erik Learned-Miller. Unsupervised joint alignment of complex images. In *ICCV*, 2007.

[5] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[6] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[7] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[8] TP Krasulina. A method of stochastic approximation for the determination of the least eigenvalue of a symmetric matrix. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 9(6):1383–1387, 1969.

[9] Gary B. Huang Erik Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014.

[10] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[11] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[12] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. Memory limited, streaming pca. In *Advances in Neural Information Processing Systems*, pages 2886–2894, 2013.

[13] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[14] Erkki Oja. Subspace methods of pattern recognition. In *Pattern recognition and image processing series*, volume 6. Research Studies Press, 1983.

[15] Cengiz Pehlevan, Anirvan M Sengupta, and Dmitri B Chklovskii. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural computation*, 30 (1):84–124, 2018.

[16] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[17] Juyang Weng, Yilu Zhang, and Wey-Shiuan Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[18] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.